Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 1998 Proceedings

Americas Conference on Information Systems (AMCIS)

December 1998

Using Divide-and-Conquer to Solve the Multiple Discrete Resource Allocation Problem

Benjamin Shao
State University of New York at Buffalo

Raghav Rao
State University of New York at Buffalo

Using Divide-and-Conquer to Solve the Multiple Discrete Resource Allocation Problem

Benjamin B. M. Shao
H. Raghav Rao
Department of Management Science and Systems
State University of New York at Buffalo

Abstract

The multiple discrete resource allocation problem (MDRAP) explores how the decision maker allocates a number of resources of different types among agents in order to achieve the aggregate maximum utility. The MDRAP belongs in the NP-hard category of time complexity, which requires excessive efforts to obtain the optimal solution even for a moderate problem size. Partial enumeration techniques such as dynamic programming and branch-and-bound are available to tackle this complexity issue to some degree. In this paper, a new partial enumeration method based on divide-and-conquer is proposed. The pronounced distinction of this divide-and-conquer approach lies in its potential ability to parallelize the solving process, and hence can obtain the optimal solution more quickly. A simulation study on a dedicated computer is conducted and presented.

Introduction

The resource allocation problem (RAP) is defined as one in which the decision maker tries to allocate a limited amount of resources among agents in order to optimize a given objective function. The RAP is a classic decision problem which appears across different disciplines including economics, computer science, artificial intelligence, management science, operations research, and others. In the context of AI, agents can be viewed as any intelligent systems (Russell and Wefald, 1991). The resources to be allocated can be divisible (continuous), like petroleum, gas, and water. Or they can be indivisible (discrete), such as trucks, workers, and machines. If there is more than one type of resource being considered, the RAP is referred to as a multiple RAP.

A multiple discrete RAP (MDRAP) in its general form is much more difficult than its continuous counterpart, and is classified as an NP-hard problem (Ibaraki 1981). An MDRAP can be formulated as an integer programming model:

P1: Maximize $\sum_{j=1}^{n} f_j(x_{1j}, x_{2j}, ..., x_{mj})$

s.t. $\sum_{j=1}^{n} x_{ij} = N_i, \qquad i = 1, 2, ..., m,$

x_{ij}: nonnegative integer, $j = 1, 2, ..., n,$

N_i: positive integer,

where N_i is the amount available for resource i and x_{ij} is the integer amount (decision variable) of resource i allocated to agent j. The objective function is the sum of individual agent's utility function f_j and represents the aggregate utility function to be maximized. It is noted that P1 is the general MDRAP and hence agent's utility function f_j does not have any restrictions like quasi-concavity, monotonicity and others.

Because an MDRAP of P1 is NP-hard, exponential time is currently needed to find an optimal solution in relation to its problem size. It means even for an MDRAP with a moderate size, excessive efforts are expected in order to find the optimal solution, especially if it is solved using the exhaustive enumeration which simply checks all possible combinations of decision variables and then identifies the particular combination as the optimal solution.

Partial enumeration techniques such as dynamic programming (Ibaraki and Katoh 1988) and branch-and-bound (Mjelde 1978 and Ibaraki 1988) are usually adopted to reduce the number of operations needed in exhaustive enumeration. In this paper, based on the idea of *divide-and-conquer,* a new partial enumeration approach is presented to solve the MDRAP. The new method has the same time complexity as those of dynamic programming and branch-and-bound, but it shows one novel advantage over

them in its potential parallelism. In other words, it is possible to apply this divide-and-conquer method on a parallel computer with a number of processors and hence to solve the MDRAP more quickly.

The New Approach

Divide-and-conquer is a strategy to split a large-scale difficult problem into a number of easier subproblems. The subproblems are solved first and then these subsolutions are combined in some way into an optimal solution of the whole (Horowitz and Sahni 1978).

The underlying idea of this new approach is to divide the original MDRAP into a number of subproblems where *only two agents are involved.* When there are only two agents to be considered for resource allocation, the solution is much easier to obtain because the amounts of resources one agent attains are the complements of those allocated to the other agent. By doing so, we do not have to worry about the possible combinations of resource bundles among all the agents, which is the primary cause for the intractability of the MDRAP.

The new divide-and-conquer approach to MDRAP is presented as follows. Subroutine *ReqMem* requests a segment of memory space with the specified size. Point variable f_r stores the beginning address of the memory segment requested by *ReqMem.* It should be noted that $f_j (j = 1, ..., n)$ is the original utility function for agent j and f_r ($r = n+1, ..., 2n-1$) represents the extra requested memory space. Vector $\mathbf{Z} = (N_1, ..., N_m)$ and vector $\mathbf{0}$ is the m-tuple of zeros.

Algorithm *Divide-and-Conquer for MDRAP*
Input: utility function f_j for agents j (j=1, ..., n).
Output: maximum aggregate utility and optimal allocation of $\boldsymbol{Z} = (N_1, ..., N_m)$.
Set $r = n$, $base = 0$;
For $I = 1$ to $\log(n) - 1$ do {
  For $J = 1$ to $(n / 2^{(I-1)}) - 1$ with increment 2 do **{**
    Set $r = r + 1$;

$$f_r = ReqMem\ (\prod_{i=1}^{m} (N_i + 1));$$

    for $\boldsymbol{x} = \mathbf{0}$ to $\boldsymbol{Z}$ do
      $f_r(\boldsymbol{x}) = \max\{ f_{base+J}(\boldsymbol{x_a}) + f_{base+J+1}(\boldsymbol{x_b}):$
        $\boldsymbol{x_a} + \boldsymbol{x_b} = \boldsymbol{x} \}$;
  } /* end of J */
  Set $base = base + n / 2^{(I-1)}$;
} /* end of I */
Set $r = r + 1$;

$$f_r = ReqMem(\prod_{i=1}^{m} (N_i + 1));$$

for $\boldsymbol{x} = \mathbf{0}$ to $\boldsymbol{Z}$ do $f_r(\boldsymbol{x}) = \max\{ f_{base+J}(\boldsymbol{x_a}) + f_{base+J+1}(\boldsymbol{x_b}):$
  $\boldsymbol{x_a} + \boldsymbol{x_b} = \boldsymbol{x} \}$;

The optimal value is $\max\{f_r(\boldsymbol{x}): \boldsymbol{x} = \mathbf{0}$ to $\boldsymbol{Z}\}$ and the optimal solution $\boldsymbol{x^*} = (\boldsymbol{x^*_1}, ..., \boldsymbol{x^*_n})$ can be traced back from f_r to each f_j (j = 1, ..., n).

Shao and Rao (1996) have shown the time complexity for this algorithm *Divide-and-Conquer for MDRAP* is $O(n\prod_{i=1}^{m} (N_i + 1)(N_i + 2))$ and the extra requested space requirement is $(n-1)\prod_{i=1}^{m} (N_i + 1)$. Both are the same as those in dynamic programming and branch-and-bound (Mjelde 1978, Ibaraki 1988, and Ibaraki and Katoh 1988). Next some performance features of this new divide-and-conquer approach are discussed.

Performance Features

Though still exponential in running time, its time complexity is as good as those obtained from dynamic programming and branch-and-bound. On the other hand, it is interesting since it is, to our knowledge, the first approach based on divide-and-conquer for solving the NP-hard MDRAP.

The number of agents, n, is assumed to be the power of two in the algorithm. This assumption can be relaxed to accommodate any number of agents and cause no trouble for practical applications due to two observations (Shao and Rao 1996). First, the algorithm intrinsically can handle the case with the number of agents equal to any power of two. Second, an optimal

solution obtained currently can be viewed as a utility function and treated as the input for the subsequent application of the algorithm.

It is observed that this new approach gives us the potential to utilize parallel processing to solve the MDRAP more quickly. In recent years, the technique of parallel processing has been used as a powerful tool to accelerate solving the complex and difficult problems, in particular NP-hard problems *per se*. By its divide-and-conquer nature, this algorithm provides potential parallelism which can solve the MDRAP in parallel. The reason is the For loop with index J can be executed in parallel on different processors, since the f_r's which the J loop works on each time are independent and disjoint. In addition, the workloads (the number of operations) distributed over the processors will be balanced, so the issues of synchronization and idle waiting frequently encountered in parallel processing (e.g., Abali et al. 1993) are not present in this algorithm.

This parallelism feature provides the divide-and-conquer approach with promising merits for solving the MDRAP. Dynamic programming and branch-and-bound are basically sequential approaches, which hence can only be implemented on one-processor computers. Without considering the communications among the processors, the parallel version of this divide-and-conquer algorithm will solve the MDRAP in time $O(\log(n)\prod_{j=1}^{m}(N_i+1)(N_i+2))$ time, which is particularly helpful when there are a large number of agents involved in the MDRAP.

Experiment and Discussion

To further investigate the performance of the new divide-and-conquer method, a simulation study is carried out on a personal computer with a 90mHz Pentium CPU. The programs for both divide-and-conquer and dynamic programming are coded in Turbo C++. The experiments are conducted for one type and two types of resources, respectively. The results are presented in Tables 1 and 2.

Table 1. Experiment Results for One Type of Resources

n	N_1	add. #	D-C	DP
4	3200	10252803	14.39	11.81
8	1600	7696007	10.88	8.18
16	800	4497615	6.42	4.67
32	400	2418431	3.41	2.53
64	200	1258863	1.86	1.32
128	100	649127	0.93	0.72
256	50	336855	0.50	0.33
512	25	179036	0.28	0.22

Table 2. Experiment Results for Two Types of Resources

n	N_1	N_2	add. #	D-C	DP
4	55	55	5097568	10.93	8.95
8	40	40	4449607	9.67	7.14
16	28	28	2649991	5.88	4.12
32	20	20	1601271	3.62	2.53
64	14	14	893025	2.04	1.43
128	9	9	381250	0.87	0.66
256	6	6	199185	0.50	0.38

The number of resources (N_1 and N_2) is determined based on the maximum array size allowed in the compiler. The numbers of additions needed for both divide-and-conquer and dynamic programming are the same, as expected and explained earlier. The running times are shown in seconds for both divide-and-conquer (D-C) and dynamic programming (DP).

As shown, when executed on a dedicated one-processor computer, the divide-and-conquer algorithm needs the same number of additions as dynamic programming, but takes a little more time than dynamic programming. The reason is because the new algorithm executes arithmetic operations of logarithm (i.e., log$n$) and exponential (i.e., $2^{(I-1)}$) to determine the upper limits for the I and J loops, respectively, while the dynamic programming sequentially determines the stages corresponding to the agent numberings without involving computations.

A parallel implementation of the divide-and-conquer approach is currently under construction on a 32-node nCUBE/2 hypercube computer so as to show its merits of parallelism over dynamic programming.

Conclusion

In this paper, a new approach based on divide-and conquer is proposed to solve the NP-hard MDRAP. Compared with the existing partial enumeration techniques such as dynamic programming and branch-and-bound, this method requires the same number of addition operations to find the optimal solution. However, the novel feature for this new approach lies in its potential parallelism which can be executed in parallel on different processors of a parallel computer. A follow-up simulation experiment is under way and will manifest this advantage when there are a large number of agents.

Acknowledgements

This research has been funded by NSF under grant #IRC 950579.

References

Abali, B., F. Ozguner, and A. Bataineh, "Balanced Parallel Sort on Hypercube Multiprocessors," IEEE Trans Parallel and Distributed Systems, 4, 1993, pp. 572-581.

Horowitz, E. and S. Sahni, Fundamentals of Computer Algorithms, Computer Science Press, Inc., MD, 1978.

Ibaraki, T., Enumerative Approaches to Combinatorial Optimization, Basel: J. C. Baltzer, 1988.

Ibaraki, T. and N. Katoh, Resource Allocation Problems, The MIT Press, Cambridge, MA, 1988.

Mjelde, K. M., "Discrete Resource Allocation by a Branch and Bound Method," J. Oper. Res. Society, 29, 1978, pp. 1021-1023.

Russell, S. and E. Wefald, Do the Right Thing: Studies in Limited Rationality. The MIT Press, MA (1991).

Shao, B. and H. R. Rao, "A Decision Model for Multiple Discrete Resource Allocation," Working Paper, Department of Management Science and Systems, SUNY at Buffalo, 1996.